THE R PROGRAMMING LANGUAGE AS A UNIFIED ENVIRONMENT FOR DATA SONIFICATION

Ethan Brown

Statisfactions: The Sounds of Data and Whimsy
http://statisfactions.wordpress.com/
ethancbrown+rforge@gmail.com

ABSTRACT

This short article explores the statistical programming environment R as a sonification interface using the add-on package `playitbyr`, currently in a pre-alpha stage of development. R's growing and vibrant community of users is a promising audience for sonification tools, and sonification can provide a welcome addition to R's arsenal of tools for making sense of multidimensional data. `playitbyr` attempts to provide intuitive functions and data structures for mapping data to sonic parameters, using MIDI or Csound for synthesis.

1. R'S POTENTIAL AS A SONIFICATION PLATFORM

Why should auditory display researchers and programmers care about R? R [1] is a free and open source programming language geared towards statistics and data analysis, allowing the analyst to work with data and simulations interactively. In Muenchen's [2] recent 2011 estimates, its academic popularity is only eclipsed by the proprietary statistical software giants SAS, SPSS, and Stata. A 2009 New York Times article [3] highlights the growing use of R in companies such as Pfizer, Google, Shell, and Bank of America, while the New York Times Graphics Department itself uses R to develop visualizations [4]. R has a large and active user base with an exponentially growing number of add-on packages and more blogs and mailing list activity than any other statistical package [2].

Sonification fits well into the R community's penchant for experimentation and trying new techniques, yet disappointingly few implementations currently exist. In 2002, Heymann and Hanson [5] sonified the Hybrid Monte Carlo algorithm and developed new tools for working with sound in R, but neither of the authors nor the general R community have followed up on this. This is not surprising, given the lack of convenient tools accessible from within an R session that allow high-level manipulation or mapping of sound. Neuwirth [6] provides an initial attempt of using Csound from R with his `Rcsound` package. However, this provides just a few basic functions for sonifying data. It has not been updated since 2004 and internet searches for the package only come up with a single publication and Neuwirth's own mailing-list comments, suggesting little interest.

The experimental `playitbyr` add-on package for R attempts to fill this gap. `playitbyr` provides a high-level interface from the R command line to map data and statistical summaries to sonic parameters, with its syntax modeled after a popular graphics add-on package called `ggplot2` [7]. Realizations are currently provided through simple MIDI and Csound interfaces. To take advantage of all features of the package, the user should install the public-domain program `csvmidi`, available at http://www.fourmilab.ch/webtools/midicsv/, a media player with basic General MIDI support, and the Csound programming language, available from http://www.csounds.com.

Currently in a pre-alpha stage of development, the package can be downloaded from the project web site at http://playitbyr.r-forge.r-project.org/ or installed directly in an R session with the command:

```
install.packages("playitbyr",
     repos="http://r-forge.r-project.org")
```

Then, the package can be loaded:

```
require(playitbyr)
```

The package is young and under active development, so the syntax and functionality of the package may change. `playitbyr` is free software licensed under the GNU Public License version 2 or greater, with source code, bug reporting, contributor resources, and more documentation available from the project web site. A related blog, *Statisfactions: The Sounds of Data and Whimsy*, has several examples of exploratory data analysis via sound.

2. STRATEGY AND IMPLEMENTATION OF `PLAYITBYR`

2.1. User Interface and Syntax

The package's central function is `sonify()`, which takes the inputs to the sonification: data, the desired mappings of constants and data variables onto audio parameters, the desired scales and scaling functions, and the desired method of rendering ("MIDI", or "csound" are currently supported). This outputs a `sonify` object which the user can subsequently manipulate, and render to sound.

For instance:

```
x <- sonify(data=iris,
      sonaes(time="Petal.Width",
      pitch = "Petal.Length",
      timbre=1, dur=1, vol=0.75),
      rendering="MIDI")
```

This code creates a `sonify` object and assigns it (with the "`<-`" assignment operator), using the famous `iris` dataset [8] which contains fifty measurements of 3 iris species in centimeters. The `sonaes` command, or SONic AESthetic, maps sonic parameters onto either constants or variables included in the data object.

Here, "Sepal.Width" is mapped onto time, and "Sepal.Length" onto pitch. The "timbre" parameter is set to 1 (representing an acoustic piano sound in General MIDI), the duration is set to 1 beat, and the volume is set to 75 percent. `playitbyr` also supports mapping onto tempo (parameter "tempo") and on to the sound balance between stereo speakers (parameter "pan"). Finally, we see that this sample will be rendered using MIDI.

R still needs more information in order to create sound. We need to specify the range of the musical parameter that we're mapping. Also, the contour of the sound object to be rendered is unclear: will the sound be rendered as distinct notes where each note represents a row in the data, or some other structure? Will the notes be connected by linear interpolation? The `playitbyr` package refers to this property as the *shape* of the sound.

With the addition of these parameters to `x`, our `sonify` object, R can now render the sound:

```
x <- (x + shape_notes()
   + scale_time_linear(0, 4)
   + scale_pitch_linear(3, 9))
x
```

This can be heard in the file *ex1.mp3*. `shape_notes()` here creates a new layer with shape "notes". (This is the only shape currently supported, but hopefully many more will become available over time.) It can take arguments with new data and different mappings; when, as here, it is left blank, R will simply use the mappings and data provided in the overall object. We can also see that "Sepal.Width" will be linearly scaled onto the times from 0 to 4 seconds, and "Sepal.Length" will be linearly scaled onto pitches from 3 to 9, where 8 equals middle C, and 1 represents an octave. This is following a notation used by Csound scores (using the `cpsoct` opcode).

Users of the celebrated `ggplot2` visualization package will find this syntax familiar. Below is the code for an analogous graph of this same data:

```
g <- ggplot(data=iris,
            aes(x=Petal.Width,
                y=Petal.Length))
g + geom_point()
```

Instead of "shape", `ggplot2` has "geom" (for "geometry"), and you can see that the code is substantially less verbose than `playitbyr`'s current syntax. `playitbyr` has fewer defaults and convenience functions, especially for scaling; see the section "Limitations and Future Plans" of this article for more discussion of this drawback.

`sonify` objects can be manipulated and changed. For instance, continuing the example above, we could choose to add or change a mapping:

```
y <- (x + sonaes(dur="Sepal.Width",
     vol="Sepal.Length",
     timbre="sine")
   + scale_dur_linear(0.5, 10)
   + scale_vol_linear(0, 0.2)
   + scale_time_linear(0, 10)
   + scale_pitch_linear(7, 12)
   + rendering("csound"))
y
```

The output of this can be heard in *ex2.mp3*. Here, "Sepal.Width" maps onto duration and "Sepal.Length" onto volume, and the timbre is now a sine wave rendered by Csound. We also defined scalings of duration and volume and redefined the scalings of time and pitch. The user can do all this based on the previous `sonify` object, and can refine the desired parameters without needing to retype the commands that created the original object (here, `x`). Note that in this case, volume is a particularly uninformative mapping, since many points are so clustered together on the time variable, "Petal.Width."

Another way users can interact with `sonify` objects is with "%+%" operator, which replaces the existing data with a new data. The variable names in the new data set need to match the ones in the old one in order to preserve the mappings created. For instance, we can easily just filter our dataset and only listen to the *versicolor* species of iris:

```
irissample <- iris[iris$Species
                   %in% "versicolor",]
y %+% irissample
```

This outputs the sound heard in *ex3.mp3*. `sonify` objects have slots that allow for multiple "layers" of the sonification, each of which can have its own shape, mapping, data, et cetera; this feature is not yet mature but should provide a wide range of flexibility for sonification. Other planned features include the computation and sonification of statistical summaries.

2.2. Interacting with MIDI files and Csound

`playitbyr` offers novel functionality, but the sound generation is done through simple interfaces to General MIDI files and Csound.

For MIDI files, `playitbyr` writes a temporary text file in comma-separated value (CSV) format with all the MIDI instructions and data. Then, it calls the public domain command line utility `csvmidi` to encode a MIDI file based on the data in the CSV file. Most operating systems come with some sort of general MIDI player, and the user can set options to tell `playitbyr` which one to use to open the file. This approach works, but can be sluggish; users are encouraged to use a command-line MIDI player such as `timidity` or `fluidsynth` to avoid delays from waiting for unnecessary graphical interfaces. See the section "Limitations and Future Plans" for more discussion of this issue and future development plans around MIDI.

`playitbyr` has a somewhat nicer interface to Csound. R has a package `tcltk`, included in most installations, which provides a mature interface to the Tcl programming language and interpreter. In turn, Csound often comes bundled with Tclcsound, which allows Tcl to control Csound in real time. Endemic to Csound is that the setup of the sound-generating instruments cannot be changed in real time, but the interface does allow the sending of control data (pitches, start times, etc.) directly to a Csound process. This is still more indirect than necessary: Tclcsound is really just a wrapper for Csound's C API. R has excellent built-in C support, so `playitbyr` could realize efficiency gains and remove an unnecessary dependency by using C instead of Tcl.

The sound programming language SuperCollider does allow a fuller range of real-time manipulation than Csound, so `playitbyr` may include an SuperCollider interface in future versions.

3. LIMITATIONS AND FUTURE PLANS

`playitbyr` faces several major limitations. Mimicking the syntax of a graphics package may encourage users to draw inappropriate analogies between visualization and sonification, a problem often discussed in the sonification literature. Furthermore, no experienced sonification researchers are currently participating in this new project. As Kramer [9] pointed out long ago, what seems an intuitive parameter mapping to a researcher may be nearly useless in terms of people's ability to extract quantitative information from the sound. Many of the package's users may have no experience with sonification, so the package ideally should provide sensible defaults and resources to help produce clear and useful auditory displays. The present lack of good defaults makes `playitbyr` daunting in the number of required options users have to specify, and also makes it excessively verbose in syntax. The project thus enthusiastically welcomes contributions and feedback from sonification veterans.

The package also faces usability issues resulting from its simplistic interface with MIDI files. `playitbyr`'s dependence on `csvmidi` currently means that the user must install the program themselves. This requires compiling `csvmidi` from source for Linux/Unix/OSX machines, and some configuration is required for it to work well with R on Windows systems. Also, one must then set up an external program to actually play the files. Creating easy and painless MIDI support in R, including writing MIDI files and sending data to MIDI ports, is thus a high priority of the project. Despite its many drawbacks, MIDI's ubiquity and familiarity still make it a compelling option for many users, especially those with no inclination to get deep into Csound.

The Csound interface faces other hurdles, however. In all likelihood, few R users are familiar enough with Csound to create their own sonification software instruments from scratch. Brave `playitbyr` users who take to the internet to find other user's instruments to use in their sonification will soon find a baffling array of designs and styles of documentation. This heterogeneity steepens the learning curve to using an instrument for sonification: `playitbyr` needs to the acceptable value ranges for any given parameter. Also, many Csound instruments that one stumbles across are quite unsuitable for sonification, whatever their musical merits. The sonification neophyte may encounter significant frustration with all these complexities.

Thus, ensuring a healthy variety of useful and portable sonification instruments is another aim of the project. One possibility would be to create a standardized format of Csound instruments and metadata. Sonifiers could submit their instruments in this format to an online repository of instruments (or to another R package). When another user downloads the instrument, R could then use this metadata to present defaults and parameter ranges for the second sonifier to work from.

4. CONCLUSION

There's still a long journey ahead for `playitbyr` and for capitalizing on R's flexibility, extensibility, and community. The muted reaction to previous sonification tools in R indicate that it will take some evangelism and better implementation to boost sonification's prominence among R users. But the joys, surprises, and usefulness of this still-novel technique could well prove a delightful boon to R's many data enthusiasts.

5. REFERENCES

[1] R Development Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2011, ISBN 3-900051-07-0. [Online]. Available: http://www.R-project.org

[2] R. A. Muenchen, "The popularity of data analysis software," Mar. 2011. [Online]. Available: http://sites.google.com/site/r4statistics/popularity

[3] A. Vance, "Data analysts captivated by r's power," Jan. 2009. [Online]. Available: http://www.nytimes.com/2009/01/07/technology/business-computing/07program.html

[4] A. Cox, "R at the new york times graphics department," AOL HQ - 770 Broadway 6th Floor New York, NY, Feb. 2011. [Online]. Available: http://vimeo.com/21020824

[5] M. Heymann and M. Hansen, "A new set of sound commands for R; sonification of the HMC algorithm," *ASA Proceedings, Statistical Computing Section*, pp. 1439–1443, 2002. [Online]. Available: http://www.amstat.org/sections/SRMS/proceedings/y2002/Files/JSM2002-000776.pdf

[6] E. Neuwirth, "R sings-or using R to sonify data," in *Proceedings of DSC*. Citeseer, 2001, p. 2.

[7] H. Wickham, *ggplot2: elegant graphics for data analysis*. Springer New York, 2009. [Online]. Available: http://had.co.nz/ggplot2/book

[8] E. Anderson, "The irises of the gaspe peninsula," *Bulletin of the American Iris Society*, pp. 2–5, 1935.

[9] G. Kramer, "An introduction to auditory display," in *Auditory Display: Sonification, Audification, and Auditory Interfaces*, ser. SFI Studies in the Sciences of Complexity, G. Kramer, Ed. Addison-Wesley, 1994, vol. XVII, pp. 1–77.